Outermedia Span Query Parser Plugin
Version 1

Datum: 19. Jun 2015
Verfasser: Andreas Koch

Inhaltsverzeichnis

1 Beschreibung

Im Unterschied zum QueryParser von Lucene (im Folgenden "LuceneQueryParser") kann der outermedia QueryParser SpanQuerys verarbeiten.

SpanQuerys sind Anfragen, die sich auf Bereiche eines Dokumentes beziehen. So kann man etwa nach einem Vorkommen von "George Bush" innerhalb von 10 Wörtern suchen, in dem kein "W" enthalten sein soll (siehe "Exclude-Query" weiter unten). SpanQuerys zu berechnen, ist relativ aufwändig.

Das outermedia SpanQuery-ParserPlugin unterstützt Unicode-Zeichen.

2 Voraussetzung

Das SpanQuery-ParserPlugin ist für Lucene 3.6 angepasst und kann nur mit dieser Version benutzt werden.

SpanQuerys können nur auf Feldern ausgeführt werden, deren Typ `TextField` ist (auf `StrField`-Feldern funktioniert das nicht).
Dies muss bei der Index-Schemadefinition beachtet werden.

2.1 Installation

Damit Solr den outermediaSpanQueryParser nutzen kann, muss die Programmbibliothek `sma-solr.jar` für Solr erreichbar und in Solr bekannt sein. Dazu könnte sie direkt in das `./lib`-Verzeichnis der Solr-Installation kopiert werden. Bei einem Update von Solr könnte dies jedoch zu Problemen führen. Wir empfehlen daher, sie in einem separaten Ordner, z.B. `./contrib/smartsearch/` abzulegen. Um diesen Ordner in Solr einzubinden, muss in der Konfigurationsdatei `solr-config.xml` ein Eintrag ergänzt werden:

```
<lib dir="../../contrib/smartsearch" />
```

2.2 Satz- und ParagraphTrenner

Intern wird anhand folgender Zeichenketten die Einschränkung auf Satz und Paragraphen beachtet.:

Token	Zeichenkette
Satztrenner	%$$%
Paragraphtrenner	$%%$

Diese Zeichen müssen daher im zu durchsuchenden Text vorhanden sein, damit der Erkennung dieser Grenzen durch dieses Plug-In gewährleistet werden kann.

2.3 Sprache

Dieses Plug-In unterstützt alle Dokumente, die das UTF-8 Encoding verwenden.

2.4 Konfiguration

Wenn der Parser installiert ist, muss das System instruiert werden, den Parser zu nutzen. Es gibt zwei Möglichkeiten, den Parser zu verwenden: als Plug-In, das bei Bedarf benutzt wird, oder als Standrad-Parser für alle Anfragen.

2.4.1 Parser als Plug-In bei Bedarf

Um ein Parser-Plug-In zu verwenden kann es jeder Zeit als Anfrage-Parameter angegeben werden: {!omspan}title:Spiderman, wobei "omspan" der Name ist, der zum Eintrag in der Konfigurationsdatei `solr-config.xml` passen muss. Dort muss ein Eintrag wiederum mit dem gleichen Namen ergänzt werden:

```
<queryParser name="omspan" class="om.solr.search.OmSpanParser" />
```

2.4.2 Parser als Standard-Parser

Der outermediaSpanQueryParser kann auch als Standard-Parser eingerichtet werden. Anfragen ohne expliziten Parser werden dann vom outermediaSpanQueryParser bearbeitet.

In der Konfigurationsdatei `solr-config.xml` gibt es einen Eintrag `<requestHandler name="search" default="true">`, der wie folgt angepasst werden muss:

```
<requestHandler name="search" class="solr.SearchHandler" default="true">
    <!-- default values for query parameters can be specified, these
         will be overridden by parameters in the request
      -->
    <lst name="defaults">
      <str name="defType">omspan</str>
      <int name="sentenceLength">200</int>
      <int name="paragraphLength">1000</int>
    </lst>
    ....
```

Die Werte für `sentenceLength` und `paragraphLength` geben die maximale Länge für SENTENCE- und PARAGRAPH-Anfragen an (siehe ?Nähe-Verknüpfungen).

2.4.3 Highlighting

Das Highlighting muss angepasst werden, so dass die speziellen Token (Satz- und Paragraphtrenner) wieder entfernt werden:
Dazu ist für die Solr Highlighting-Komponente ein spezieller Encoder zu verwenden. Die Konfiguration erfolgt in solrconfig.xml.

```
<searchComponent class="solr.HighlightComponent" name="highlight">
    <highlighting>
    ...
    <!-- Configure the filtering encoder -->
    <encoder name="filteredDefault"
class="om.solr.highlight.FilteringDefaultEncoder">
        <lst name="defaults">
            <str name="hl.filterToken.1"><![CDATA[$%%$]]></str>
            <str name="hl.filterToken.2"><![CDATA[%$$%]]></str>
        </lst>
    </encoder>
    <encoder name="filteredhtml" default="true"
```

```
class="om.solr.highlight.FilteringHtmlEncoder">
        <lst name="defaults">
            <str name="hl.filterToken.1"><![CDATA[$%%$]]></str>
            <str name="hl.filterToken.2"><![CDATA[%$$%]]></str>
        </lst>
    </encoder>
    ....
    </highlighting>
</searchComponent>
```

Durch default="true" kann gesteuert werden, welcher der filternden Encoder verwendet werden soll. Der FilteringDefaultEncoder basiert auf dem Solr-DefaultEncoder und der FilteringHtmlEncoder auf dem Solr-HtmlEncoder. Alle bei hl.filterToken.X aufgelisteten Token werden entfernt. Die Nummerierung muss bei 1 beginnen, und die Verarbeitung endet mit dem ersten nicht gefundenen Parameter (im Beispiel oben 3).

3 Benutzung

Benutzeranfragen werden vom System in Terme, Operatoren und Klauseln zerlegt und dann ausgewertet. Im Folgenden werden zunächst diese Begriffe kurz erklärt und dann Beispielanfragen erläutert.

3.1 Terme und Felder

Ein **Term** ist (meistens) ein Wort, z.B. "Peter". Jede Anfrage muss einen oder mehrere Term(-e) enthalten, nach denen gesucht werden kann.

SmartSearch speichert für Dokumente verschiedene **Felder**, z.B. "titel" oder "text". Um Terme innerhalb eines bestimmten Feldes zu suchen, muss deren Name von einem Doppelpunkt "**:**"gefolgt angegeben werden, z.B. titel **:** "Spiderman". Eine Suche kann sich auch auf mehrere Felder beziehen, z.B. titel:"Spiderman" AND text:"Goblin".

SmartSearch nutzt standardmäßig das Feld "text". Die Anfrage **title:**green goblin würde als **title:**"green" AND **text:**"goblin" interpretiert werden, was möglicherweise ungewollt ist.

3.1.1 Anführungszeichen

Gehören mehrere Wörter zusammen, können sie durch Anführungszeichen zu einem Term zusammengefasst werden, z.B. **"**Peter Parker**"**.

3.1.2 Platzhalter

Ein Fragezeichen steht für einen beliebigen Buchstaben. So würde H**?**nd zu Dokumenten mit den Wörtern "Hand" einerseits oder "Hund" andererseits passen.

Ein Stern steht für beliebig viele Buchstaben. Von Test***** würden die Wörter "Test", "Tester", "Testbetrieb" und viele mehr erfasst werden.

3.1.3 Bereiche

Bereiche (engl. range) werden - wie in Lucene - mit **[***<Anfang>* **TO** *<Ende>***]** spezifiziert.

Beispiel:

- jahr:[2001 TO 2011] sucht im Feld "jahr" nach Werten 2001, 2002, ... 2011. Mit einer Bereichsklausel lassen sich Suchen eingrenzen.

3.1.4 Verstärkung

Liefert eine Anfrage mehrere Treffer, werden die Dokumente standardmäßig nach Häufigkeit der Treffer sortiert. Dokumente, in denen die gesuchten Terme häufiger vorkommen (oder Dokumente, die kürzer sind, siehe term frequency--inverse document frequency), werden als relevanter erachtet. Einzelne Terme (und/oder Teilfragen) können - wie in Lucene - mit <Term>^1.5 schwerer gewichtet (engl. boost) werden. Bei der Suche title:Spiderman^2.0 OR text:Spiderman, werden Dokumente, bei denen "Spiderman" im Titel vorkommt, als 2-mal so relevant erachtet, wie Dokumente, bei denen "Spiderman" im Text vorkommt.

Als Wert können alle Zahlen mit genau einer Kommastelle verwendet werden. Terme (und Klauseln) ohne Boost werden intern mit dem Standardwert ^1.0 verarbeitet.

3.1.5 Unscharfe Begriffe

Einzelne Terme können durch eine Tilde "~" mit einer gewissen Unschärfe (engl. fuzzy) angegeben werden. Als Parameter wird eine Zahl zwischen 0.0 (sehr ungenau) und 1.0 (exakt) erwartet. FuzzyQuerys funktionieren nicht zusammen mit Anführungszeichen.

3.1.6 Suche nach allen Dokumenten

Die Suche nach allen Werten in allen Feldern lautet:

```
*:*
```

3.2 Operatoren

Terme (und Klauseln) können mit **Operatoren** verknüpft werden. Das Resultat einer Verknüpfung sind **Klauseln** (also Teilanfragen). Um Mehrdeutigkeit zu verhindern, können Klauseln in runden **Klammern** zusammengefasst werden.

Es gibt Operatoren, die sich auf den folgenden Term beziehen, z.B. NOT. Die meisten Operatoren beziehen sich auf die beiden Terme direkt vor und direkt hinter ihnen. (z.B. AND, OR, NEAR)

3.2.1 Und-Verknüpfung

AND verknüpft zwei Terme (oder Klauseln), die beide im Dokument vorkommen müssen. AND kann alternativ auch als **&&** geschrieben werden.

Mehrere AND-Klauseln können direkt hintereinander geschrieben werden.

Beispiele:

- Queen **AND** Marry für eine Suche nach Dokumenten, in denen sowohl "Queen" als auch "Marry" vorkommen.
- "Queen Marry" **AND** "Pazifischer Ozean" für eine Suche nach den beiden Wörtern "Queen Marry" (die direkt hintereinander stehen müssen) und den beiden Wörtern "Pazifischer Ozean" (die ebenfalls direkt hintereinander stehen müssen). Wo die beiden Wortgruppen wiederum im Dokument vorkommen ist unwichtig -- sie müssen nur beide vorkommen.
- Queen **AND** Marry **AND** pazifi* **AND** Ozean für eine Suche nach allen vier Wörtern (egal in welcher Reihenfolge, wobei "Pazifik" oder "pazifischer" eingeschlossen sind).

3.2.2 Oder-Verknüpfung

OR verknüpft zwei Terme (oder Klauseln), von denen mindestens einer in jedem Ergebnis vorkommen muss. OR kann alternativ auch als **||** geschrieben werden.

Mehrere OR-Klauseln können direkt hintereinander geschrieben werden.

Beispiele:

- Allianz OR HUK für eine Suche nach Dokumenten, in den Allianz oder HUK (oder beide) Terme vorkommen
- title:Allianz OR text:Allianz für eine Suche nach Dokumenten, in denen "Allianz" entweder im Titel oder im Text (oder beides) vorkommt.
- Äpfel OR Apfel OR Birne OR Birnen OR Obst für eine Suche nach einem der Wörter.

3.2.3 Nicht-Operation

NOT schließt nur Dokumente in die Ergebnisliste ein, die den angegebenen Term nicht enthalten.

Beispiel:

- Apple NOT Computer NOT iPhone NOT iPad um nach englischen Äpfeln zu suchen ohne Apple, den iPhone-Hersteller, zu finden.

Achtung: Eine NOT-Query kann nicht als Klausel einer NEAR- oder EXCLUDE-Suche verwendet werden!

3.2.4 Nähe-Verknüpfungen

Die beiden Terme (oder Klauseln) dürfen maximal einen bestimmten Abstand zueinander haben. Die Reihenfolge ist dabei egal. Intern verwendet das System sogenannte SpanQuerys. Diese dürfen wiederum keine Negationen (d.h. NOT-Operatoren) enthalten, weil diese nicht als Begrenzung eines Bereiches dienen können.

Der Operator kann lang als NEAR/n oder kurz als N/n geschrieben werden, wobei n eine natürliche Zahl sein muss.

Um mehrere Terme in einem Bereich anzugeben, muss der Operator vor die Terme gesetzt und die Terme müssen in Klammern zusammen gefasst werden.

Zwei Spezialfälle von NEAR sind SENTENCE (engl. Satz) und PARAGRAPH (engl. Absatz). Bei diesen Operatoren werden die umstehenden Terme (oder Klauseln) nur innerhalb eines Satzes - oder Absatzes - gesucht.

Von Hause aus funktionieren beide Operatoren mit Sätzen und Absätzen von maximal 1.000 Wörtern Länge. Kleinere Werte führen zu schnellerer Verarbeitung, größere schließen keine eventuellen Treffer aus. Die Werte können unabhängig voneinander eingestellt werden.

Beispiele:

- Alianz NEAR/6 kauft für eine Suche nach Dokumenten, in denen "Allianz" und "kauft" egal in welcher Reihenfolge nicht mehr als sechs Wörter voneinander entfernt stehen (genaugenommen: ein Bereich mit "Allianz", "kauft" und bis zu 6 anderen Wörtern).
- Allianz SENTENCE verkauft für eine Suche nach Dokumenten, in denen "Allianz" und "verkauft" im selben Satz vorkommen.
- Merkel PARAGRAPH Kohl für eine Suche nach Dokumenten, in denen es einen Absatz mit den Worten "Merkel" und "Kohl" gibt.
- NEAR/20 (Allianz Deutschland Stellenangebot) sucht nach Dokumenten mit den Termen "Allianz", "Deutschland" und "Stellenangebot", die - egal in welcher Reihenfolge - in einem Bereich mit höchstens 20 anderen Wörtern vorkommen.
- SENTENCE (Spiderman Green Goblin) sucht nach Dokumenten, in denen "Spiderman", "Green" und "Goblin" in einem Satz genannt werden. Alternative Terme müssen mit Klammern zu Klauseln zusammengefasst werden:
- PARAGRAPH (Kinder Ernährung (Obst OR Gemüse)) sucht nach Dokumenten mit einem Absatz, in dem es um "Kinder", "Ernährung" und alternativ "Obst" oder "Gemüse" geht.

3.2.5 Ausschluß-Verknüpfung

Ausschluss-Verknüpfung, um einen Such-Term durch Ausschluss genauer zu bestimmen. Der Operator kann lang als EXCLUDE oder kurz als X geschrieben werden.

Achtung: Der zweite Term (d.h. der exclude-Teil) wird nur innerhalb des ersten Bereichs (d.h. dem include-Teil) gesucht. Eine Suche nach (Johann N/1 Bach) EXCLUDE Christian filtert Treffer mit "Christian" nur dann, wenn das Wort Christian zwischen "Johann" und "Christian" steht. Eine NOT-Query filtert alle Dokumente aus, die überhaupt das Wort "Christian" enthalten.

Beispiele:

- Peter* EXCLUDE "Peter Pan" sucht nach verschiedenen Peters, aber nicht nach "Peter Pan".
- (Allianz NEAR/3 Börse) X verkauft sucht nach Dokumenten mit Passagen, die "Allianz" und "Börse", aber nicht das Wort "verkauft" enthalten.

3.3 Verknüpfung von Klauseln

Eine Anfrage kann sich aus mehreren (auch verschachtelten) Teilfragen ("Klauseln") zusammen setzen. Dabei ist zu beachten, dass Terme SpanQuerys und SpanQuerys wiederum eine Unterform der LuceneQuerys sind:

- *LuceneQuery*
    - AND (LuceneQuery AND LuceneQuery)
    - OR (LuceneQuery OR LuceneQuery)
    - NOT (NOT LuceneQuery)
    - *:*
    - *SpanQuery*
        - NEAR (SpanQuery NEAR/6 SpanQuery)
        - SENTENCE (SpanQuery SENTENCE SpanQuery)
        - PARAGRAPH (SpanQuery PARAGRAPH SpanQuery)
        - AND (SpanQuery AND SpanQuery)
        - OR (SpanQuery OR SpanQuery)
        - EXCLUDE (SpanQuery EXCLUDE SpanQuery)
        - *Term*
            - Wort
            - "mehrere Wörter"
            - irgendw*
            - ähnlich~0.3
            - wichtig^1.5

Auf jeder Ebene können Querys niedrigerer Ebene eingebunden werden, umgekehrt nicht. Insbesondere können NOT-Querys nicht innerhalb von SpanQuerys (insbesondere NEAR und EXCLUDE) verwendet werden.

Querys können durch Klammern zusammen gefasst werden.

Mehrere (jeweils gleiche) AND- oder OR-Querys können direkt hintereinander geschrieben werden.

Beispiele:

- Erlaubt: (aa OR bb) NEAR/2 cc. Das OR wird als Span-OR interpretiert.

- Erlaubt: (aa NEAR/3 bb) NOT cc. Near-Query ist hier Teil eines Lucene-Querys, was erlaubt ist.
- Erlaubt: aa AND bb AND cc. Mehrere gleiche Operatoren können hintereinander geschrieben werden und werden von links nach rechts interpretiert.
- Erlaubt: aa OR bb OR cc. Mehrere gleiche Operatoren können hintereinander geschrieben werden (siehe oben).
- Erlaubt: title:(Franz Beckenbauer~0.4^1.5) OR text:(Beckenbauer). Im Feld "titel" nach den Wörtern "Fanz" und "Beckenbauer" suchen oder im "Text" nur nach "Beckenbauer" wobei im Titel die genaue Schreibweise nicht so wichtig und ein Treffer für "Beckenbauer" im "Titel" relevanter ist als im "Text".
- Ergebnis nicht definiert: aa OR bb AND cc OR dd. Es sollten in diesem Fall explizit Klammern gesetzt werden, z.B. (aa OR bb) AND (cc OR dd).
- Nicht möglich: (aa NOT bb) NEAR/4 cc. Ein NOT-Query kann nicht als Teil eines NearQuerys bzw. SpanQuerys verwendet werden.
- Nicht möglich: *:* EXCLUDE "new york" Ein Match-All-Documents-Query (*:*) kann nicht als Klausel in einem EXCLUDE- bzw. SpanQuery dienen. (Das ließe sich allerdings als NOT "new york" formulieren.)

3.4 Fehlerfälle und -meldungen

3.4.1 NOT in NEAR/x-Klausel (bzw. Span-Klausel)

- Beispiel: (thomas NOT müller) NEAR/2 bayern
- Meldung: Cannot interpret query '(thomas NOT müller) NEAR/2 bayern': Encountered " <NEAR_L> "NEAR/2 "" at line 1, column 20.
- Erklärung: Parser erkennt das NOT und kann dann keinen Span-Operator verarbeiten. In spitzen Klammern steht die interne Bezeichnung des Operators (noch genauer: des Parser-Tokens) in Anführungszeichen, dahinter die konkret gefundene, falsche Zeichenkette.

3.4.2 Fehlende schließende Klammer

- Beispiel: internal AND (speaker OR drive
- Meldung: Cannot interpret query 'internal AND (speaker OR drive': Encountered "<EOF>" at line 1, column 30.
- Erklärung: Parser sucht nach schließender Klammer und gerät ans Ende der Eingabe ("End of file").

3.4.3 Zu viele schließende Klammern

- Beispiel: boris AND becker) NOT werbung
- Meldung: Cannot interpret query 'boris AND becker) NOT werbung': Encountered " ")" ") "" at line 1, column 16.
- Erklärung: Parser findet unerwartet eine schließende Klammer.

3.4.4 Fehlendes Anführungszeichen

- Beispiel: "Franz Beckenbauer OR "Lothar Matthäus"
- Meldung: Cannot interpret query '"Franz Beckenbauer OR "Lothar Matthäus"': Lexical error at line 1, column 40. Encountered: <EOF> after : ""
- Erklärung: Parser sucht nach abschließendem Anführungszeichen und gerät ans Ende der Eingabe.

3.4.5 Operator ohne Term

- Beispiel: cd-rom OR OR dvd
- Meldung: Cannot interpret query 'cd-rom OR OR dvd': Encountered " <OR> "OR "" at line 1, column 10.
- Erlkärung: Parser erkennt einen Operator (hier das zweite OR), obwohl er einen Term (für das erste OR) erwartet hätte.